Remarks

This reply is responsive to the Office communication of January 26, 2006. Unless otherwise indicated, page and paragraph references are to that communication.

Claim Rejections—35 U.S.C. § 112

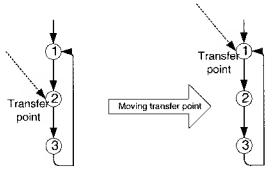
1. 35 U.S.C. § 112, First Paragraph

Claims 1-10 stand rejected under 35 U.S.C. § 112, first paragraph, as failing to comply with the enablement requirement (page 5, ¶ 9). The Examiner contends basically that the specification does not enable one of ordinary skill to locate, copy and/or move a transfer point. The Examiner contends further that while applicants' April 2004 remarks were convincing, they are vitiated by applicants' October 2005 remarks distinguishing over Aho et al., Compilers: Principles, Techniques and Tools, Chapter 10, "Code Optimization", pages 585-722 (1986) ("Aho"). Applicants respectfully disagree.

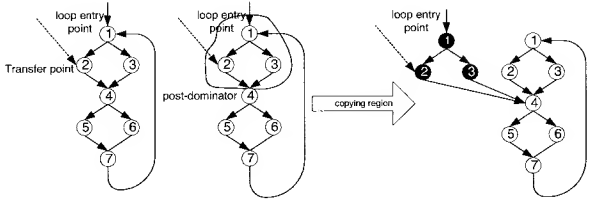
Applicants' claims define a transfer point as a point at which program execution is transferred from an interpreter process to a loop process of a compiled code process. This is consistent with the originally filed specification, which defines a transfer point as "a point whereat program execution is transferred from the Java interpreter 124 to the JIT compiler 126" (page 14, lines 29-30). Since the problematic transfer points are those to the inside of a loop (page 4, lines 10-12), it is perfectly proper to restrict the claims to such transfer points. Thus, it is not apparent why anyone would have a problem with locating, copying and/or moving such transfer points.

In the October 2005 amendment, applicants argued (page 12) that their invention as it related to the movement of a transfer point did not involve the movement of code, such as described in Aho. This is perfectly consistent with applicants' definition of a transfer point. One might "move" a transfer point by changing the target address of a transfer instruction so that it points to the top of a loop rather than inside the loop. For example, in Fig. (a) below, an instruction

formerly pointing to node 2 would be modified to point to node 1 instead, thereby moving the transfer point to the top of the loop.



(a) An example of moving transfer point



(b) An example of copying a region from entry point to the post-dominating point

Applicants further argued in the October 2005 amendment (at pages 13-14) that the flow graph depicted in Fig. 10.49 of Aho (on page 668) did not depict entry points or transfer points. On reconsideration of this question, it appears that this may be incorrect and that in Fig. 10.49(a), nodes 2 and 3 may be deemed entry points (from node 1) to the loop comprising nodes 2 and 3. Applicants therefore withdraw that aspect of their previous argument. Fig. 10.49 still doesn't show the moving or copying aspects of applicants' claimed invention, however, so the thrust of their previous argument is unaffected.

In summary, nothing in applicants' previous patentability arguments (as qualified above) impugns in any way applicants' previous arguments regarding transfer points and adequacy of disclosure. Applicants therefore respectfully request that this rejection be withdrawn.

2. 35 U.S.C. § 112, Second Paragraph

Claims 1-10 also stand rejected under 35 U.S.C. § 112, second paragraph, for failing to particularly point out and distinctly claim what applicants regard as their invention (page 6, ¶ 11), as well as for failing to set forth the subject matter that applicants regard as their invention (page 6, ¶ 12).

Again, this new ground for rejection appears to be based on remarks made in the October 2005 amendment. As noted above, applicants argued in that amendment that code motion is not involved in that aspect of their invention relating to movement of transfer points to the top of the loop. The Examiner appears to be conflating this aspect of applicants' invention, as shown in Fig. (a) above and recited in claims 1-6 and 8-9, with the copying of code from the top of a loop process, as shown in Fig. (b) above and recited in claims 1-4, 6-7 and 9-10. While the movement of transfer points typically does not involve the movement of code, the copying of code does involve the insertion of code, although even here not the movement as opposed to copying of code from elsewhere in the routine. Accordingly, for reasons similar to those set forth above, none of applicants' previous arguments in favor of patentability forms any basis for contesting what they regard as their invention.

Claim Rejections—35 U.S.C. § 103

Each of the rejected claims recites either the movement of transfer points to the top of a loop process, the copying of code from the top of a loop process, or both. Since either of these recitations is sufficient to distinguish over the art cited, it will be sufficient for the purposes of this response to address these two aspects of applicants' invention.

1. Moving Transfer Points to Top of Loop Process

This aspect of applicants' claimed invention, recited in claims 1-6 and 8-9, involves moving one or more transfer points, at which program execution is transferred from an interpreter process to a loop process of a compiled code process, to the top of the loop process if they can be moved there without a problem occurring. This is shown in Fig. (a) above.

Claims 1-6 stand rejected as being unpatentable over Aho in view of Bak et al., U.S. Patent 6,513,156 ("Bak"), Bacon et al., "Compiler Transformations for High-Performance Computing", ACM Computing Surveys, vol. 26, no. 4, pages 345-420 (Dec. 1994) ("Bacon") and Koblenz et al., U.S. Patent 5,530,866, newly cited ("Koblenz") (page 8, ¶ 14). Claims 5-6 and 8-9 also stand rejected as being unpatentable over Bak in view of Koblenz and Aho (page 13, ¶ 16). These rejections are respectfully traversed.

These rejections are both predicated on Koblenz's supposed teaching of the movement of transfer points to the top of a loop process. As the abstract states, Koblenz relates to methods for "allocating physical registers within a compiler phase to achieve efficient operation of a target CPU." The operation of this system involves in part the analysis of a "tile tree" (Fig. 1b), defined as "[a] structure of tiles representing the loop and conditional structure of the code under analysis". As the patentee elaborates at column 8, lines 15-28 (emphasis added):

> This loop structure of the code is identified on the basis of intervals in the control flow graph as accomplished in prior art processes. An interval in the control flow graph is a set of basic blocks that form a loop in the code. Intervals, like tiles and loops, nest.
>
> In a tile tree construction process useful in the practice of the present invention, a "loop top" is defined as the single basic block having incoming back edges that dominates every basic block in its loop. Irreducible loops do not exhibit a loop top; however, all basic blocks in an irreducible loop that are reached by a forward control flow edge from a basic block outside the loop can be combined into a single summary loop top in constructing the tile tree. This summary node will dominate every basic block in the loop.

The Examiner argues that the highlighted language constitutes the teaching of moving transfer points to the top of a loop process if they can be moved there without a problem occurring. as

claimed by applicants. However, even if the formation of a single summary loop top could be regarded as such, it is merely the manipulation of an abstract flow graph for the purpose of assigning physical registers, not the modification of a loop process for the purpose of code optimization, as claimed by applicants.

Accordingly, Koblenz does not teach moving one or more transfer points at which program execution is transferred to a loop process to the top of the loop process as claimed by applicants. Therefore, claims 1-6 and 8-9, which are directed to this aspect of applicants' invention, distinguish patentably over the art cited by the Examiner.

2. Copying Code from Top of Loop Process

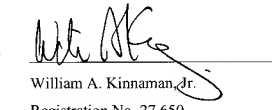
This aspect of applicants' invention, recited in claims 1-4, 6-7 and 9-10, involves copying code, extending from the top of the loop process to a point that post-dominates the top of the loop process and the transfer points, to a location immediately preceding the loop process if the transfer points are located inside the loop process. This is shown in Fig. (b) above.

All of these claims stand rejected as being unpatentable over Aho in view of Bak, Bacon and Koblenz (page 8, ¶ 14), while claims 6 and 9, as previously noted, stand rejected as being unpatentable over Bak in view of Koblenz and Aho (page 13, ¶ 16). This rejection is respectfully traversed, for reasons similar to those presented above. Whatever Koblenz's teachings in this regard, they relate merely to the manipulation of an abstract flow graph for the purpose of assigning physical registers, not the modification of a loop process for the purpose of code optimization, as claimed by applicants.

Conclusion

Reconsideration of the application is respectfully requested. It is hoped that upon such consideration, the Examiner will hold all claims allowable and pass the case to issue at an early date. Such action is earnestly solicited.

Respectfully submitted,
TOSHIAKI YASUE et al.

By 

William A. Kinnaman, Jr.
Registration No. 27,650
Phone: (845) 433-1175
Fax: (845) 432-9601

WAK/wak